09/960,630

L Number	Hits	Search Text	DB	Time stamp
1	4808	(graphics adj3 (language or application)) and @ad<19991206	USPAT; US-PGPUB	2003/10/27 15:31
2	2803	(graphics adj (language or application)) and @ad<19991206	USPAT; US-PGPUB	2003/10/27 15:31
3	505	((graphics adj (language or application)) and @ad<19991206) and (mathematical or math adj3 functions)	USPAT; US-PGPUB	2003/10/27 15:32
4	616	((graphics adj (language or application)) and @ad<19991206) and (mathematical or math or logical adj3 functions)	USPAT; US-PGPUB	2003/10/27 15:32
5	0	(((graphics adj (language or application)) and @ad<19991206) and (mathematical or math or logical adj3 functions)) and "sine operation"	USPAT; US-PGPUB	2003/10/27 15:36
6	0	(((graphics adj (language or application)) and @ad<19991206) and (mathematical or math or logical adj3 functions)) and ("sin" adj operation)	USPAT; US-PGPUB	2003/10/27 15:36
7	0	(((graphics adj (language or application)) and @ad<19991206) and (mathematical or math or logical adj3 functions)) and ("sin" adj3 operation)	USPAT; US-PGPUB	2003/10/27 15:36
8	3	(((graphics adj (language or application)) and @ad<19991206) and (mathematical or math or logical adj3 functions)) and ("sin" same operation)	USPAT; US-PGPUB	2003/10/27 16:09
9	23	(((graphics adj (language or application)) and @ad<19991206) and (mathematical or math or logical adj3 functions)) and (math same operation)	USPAT; US-PGPUB	2003/10/27 16:09
10	2	(((graphics adj (language or application)) and @ad<19991206) and (mathematical or math or logical adj3 functions)) and ((math and logical) same operation)	USPAT; US-PGPUB	2003/10/27 17:01
11	5	((graphics adj3 (language or application)) and @ad<19991206) and ((math and logical) same operation)	USPAT; US-PGPUB	2003/10/27 17:01

swizzle

To convert external names, array indices, or references within a data structure into address pointers when the data structure is brought into main memory from external storage (also called "pointer swizzling"); this may be done for speed in chasing references or to simplify code (e.g. by turning lots of name lookups into pointer dereferences). The converse operation is sometimes termed "unswizzling".

See also snap.

[Jargon File]

Try this search on OneLook / Google

Nearby terms: switched virtual connection « switching hub « switch statement « **swizzle** » swung dash » sy » Sybase, Inc.

86 citations found. Retrieving documents...

J. E. B. Moss. *Working with persistent objects: To swizzle or not to swizzle.* IEEE Transactions on Software Engineering, 18(8):657–673, Aug 1992.

This paper is cited in the following contexts:

First 50 documents Next 50

Managing Kernel Memory Resources from User Level - Haeberlen (2003) (Correct)

....a malicious principal could change them and thus cause inconsistencies in the kernel state; if it is able to guess the address of a privileged object, it could even compromise protection. **This problem can be solved with pointer swizzling, a standard technique used in persistent object stores [41].** When part of a composite object is preempted, all references to and from this part are replaced with virtual addresses in resource space, which is local to the principal. **These** addresses can then be used to locate the referenced objects when the part is restored. **Obviously**, a malicious ....

J. E. B. Moss. *Working with persistent objects: To swizzle or not to swizzle.* IEEE Transactions on Software Engineering, 18(8):657--673, Aug 1992.

Software Circulation using Sandboxed File Space -.. - Kato, Oyama, Kanda.. (Correct)

....operations can be iterated at anytime between distributed computing sites. **We achieved this mechanism, without assuming the use of special hardware, by implementing an iterative pointer relocation technique, which is referred to as pointer swizzling in the context of persistent object management [15].** For correct relocation, Planet has to know the locations of all the pointers on a mobile segment. **The** language processors of strongly typed languages can provide this location information. **Weakly** typed languages or semi strongly typed languages, such as C or C , require assistance from ....

J. E. B. Moss. *Working with persistent objects: to swizzle or not to swizzle.* IEEE Trans. Software Engineering, 18 (8):657--673, Aug. 1992.

Function-Based Indexing for Object-Oriented Databases - Hwang (1994) (3 citations) (Correct)

....Objects at the FE refer to each other by local virtual memory addresses. **When** a persistent object copy is brought to an FE, its references must be swizzled before being used. **That** is, its xrefs must be converted into virtual memory addresses. **Currently, we are looking at two forms of swizzling[50]: object swizzling, where all of the xrefs in an object are swizzled when the object is accessed for the first time, and pointer swizzling, where an individual xref is swizzled when it is used for the first time to access the object it refers to.** In object swizzling, a xref for an object that is ....

J. E. B. Moss. *Working with persistent objects: To swizzle or not to swizzle.* IEEE Transactions on Software Engineering, 18(3), August 1992.

The ANU Orthogonlly Persistent Java System - Molinari (2000) (Correct)

....spaces of the persistent object store and the AJM (processor memory) are distinct. **Efficiency** concerns require that the AJM accesses objects in terms of processor memory addresses. **A** one to one mapping needs to be managed between the two address spaces. **This notion is termed swiz zling see [8] for a discussion of the issues.** V. **CONCURRENCY** If we are only concerned with a single Java program accessing the persistent object store at a time, then the abstract arrangement of figure 3 and the pragmatic arrangement of figure 4 applies. **The** more realistic arrangement, though, is that ....

J. Eliot B. Moss, "*Working with persistent objects: To swizzle or not to swizzle*," IEEE Transactions on Software Engineering, vol. SE-18, no. 8, pp. 657-673, August 1992.

TDB: A Database System for Digital Rights Management - Vingralek, Maheshwari, Shapiro (2001) (1 citation) (Correct)

.... **Although we wanted tight integration with C , we did not want to invest the effort and incur the implementation complexity for providing transparent persistence (also known as orthogonal persistence [1] Transparent persistence includes automatic swizzling of persistent ids into memory pointers [29], automatic locking for concurrency control, and in some cases garbage collection based on reachability.** These features usually require a tool to parse, compile, or instrument object representations. **We** settled for requiring explicit application calls to lock, swizzle, and reclaim objects. **While** ....

J. Moss. *Working with persistent objects: To swizzle or not to swizzle*. IEEE Transactions on Software Engineering, 18(8), 1992.

HL$')7L$ah - Ahxz Qj Ffi (Correct)

....ae,#### 9L =K#;Q # #ae 4 # j # =K#;Q# 6Lf 3 # (af)im 4 AE # #9L### # JL ###### ffi) ae #UW =K#;Q### im #### #chhim ### =K#;Qae,#### ffi ### j . # ffi ###### lm #) ah# ### ae ae,##### ### =K#;Q # #ae # ffi ### JK#,##### ## #### ffi 4 # #j . **[Mos92, WD92] ffI =K#;Q ### j # (object naming) ###ae,# =K#;Q### ###ae 9L #)ae # ###### ### ae,#### # #2L ### HL)7L ah9L ae,#### ffi # # ahXZ# ae # #ae,#(entry point) m#j .** K#;Qj # ffi ###)ae # ##### =K#;Q # #ae ### 4 ### j #2M### ##ae ffi OE #### #ffi # i ....

J. Eliot B. Moss. "*Working with Persistent Objects: To Swizzle or Not to Swizzle*". IEEE Transactions on Software Engineering, 18(8):657--673, August 1992.

QuickStore: A High Performance Mapped Object Store - White, DeWitt (1994) (31 citations) (Correct)

.... **main memory [Wilso90] The advantage of this approach is that access to in memory persistent objects is just as efficient as access to transient objects, i.e. application programs access objects by dereferencing normal virtual memory pointers, with no overhead for software residency checks as in [Moss92, Schuh90, White92].** QuickStore is implemented as a C class library that can be linked with an application, requiring no special compiler support. **Instead**, QuickStore uses a modified version of the GNU debugger (gdb) to obtain information describing the physical layout of persistent objects. **QuickStore** uses the ....

....expected to happen often) then a search in an inmemory tree structure is required to determine the starting address of the database file. **We** next discuss previous performance studies of pointer swizzling and object faulting techniques, and point out how the study presented here differs from them. **[Moss92] contains a study of several software swizzling techniques and examines various issues relevant to pointer swizzling.** Among these are whether swizzling has better performance than simply using object identifiers to locate objects, and whether objects should be manipulated in the buffer pool of the ....

[Article contains additional citation context not shown here]

J. Eliot B. Moss, "*Working with Persistent Objects: To Swizzle or Not to Swizzle*", IEEE Transactions on Software Engineering, 18(8):657-673, August 1992. - 39 -

---

Bounded Parallel Garbage Collection: Implementation.. - Vaughan.. (Correct)

....Table (CTT. **A** simple system design may retain all pointers within objects as object identifiers, performing a translation on each pointer dereference, however significant performance improvement is gained [GR83] if the object identifier is overwritten by the in cache address. **The term swizzling [Mos92] has been adopted for the action of overwriting an object identifier with the in cache address of the referend.** Swizzling can be done eagerly, where the system ensures that all pointers in the cache are so overwritten, or lazily, where the swizzling is delayed until needed, such as the first time ....

Moss, J.E.B. "*Working with Persistent Objects: To Swizzle or Not to Swizzle*". IEEE Transactions on Software Engineering 18, 3 (1992).

HAC: Hybrid Adaptive Caching for Distributed Storage Systems - Castro, Adya, Liskov, Myers (1997) (8 citations) (Correct)

....as at the server to avoid extra copies. **It** is not practical to represent object pointers as orefs in the client cache because each pointer dereference would require a table lookup to translate the name into the object s memory location. **Therefore, clients perform pointer swizzling [KK90, Mos92, WD92] i.e. replace the orefs in objects instance variables by virtual memory pointers to speed up pointer traversals.** HAC uses indirect pointer swizzling [KK90] i.e. the oref is translated to a pointer to an entry in an indirection table and the entry points to the target object. ....

....Section 4 were obtained on a machine with 64 bit pointers. **Both** pointer swizzling and installation of objects, i.e. allocating an entry for the object in the indirection table, are performed lazily. **Pointers are swizzled the first time they are loaded from an instance variable into a register [Mos92, WD92] the extra bit in the oref is used to determine whether a pointer has been swizzled or not.** Objects are installed in the indirection table the first time a pointer to them is swizzled. **The** size of an indirection table entry is 16 bytes. **Laziness** is important because many objects fetched to ....

J. E. B. Moss. *Working with persistent objects: To swizzle or not to swizzle.* IEEE Transactions on Software Engineering, 18(3):657–673, August 1992.

Lazy Reference Counting for Transactional Storage Systems - Castro, Adya, Liskov (1997) (1 citation) (Correct)

....databases. **For** objects in the client cache, it replaces contained global object names by virtual memory pointers, thereby avoiding the need to translate the object name to a pointer every time a reference is used. **Pointer swizzling schemes can be classified as direct [13] or indirect [11, 10].** In direct schemes, swizzled pointers point directly to an object, whereas in indirect schemes swizzled pointers point to an indirection table entry that contains a pointer to the object. **Indirection** allows Hac to efficiently evict cold objects from the cache and compact hot objects to avoid ....

....an indirect form of lazy swizzling on discovery [13] references are swizzled to point to an entry in an indirection table that points to the object s fields. **Swizzling** is performed when a reference is loaded from an instance variable into a local variable. **This is implemented using edge marking [10]; references are marked as swizzled or unswizzled using one bit.** When a reference is loaded from an instance variable into a local variable, a check is performed to determine whether the reference is swizzled or not. **If** the reference is not swizzled, the object name is translated into a pointer to ....

J. Eliot B. Moss. *Working with persistent objects: To swizzle or not to swizzle.* IEEE Transactions on Software Engineering, 18(3), August 1992.

Parallel Pointer-Based Join Algorithms in Memory Mapped .. - Buhr, Goel, Nishimura, .. (1996) (1 citation) (Correct)

....be possible to locate all pointers so they can be updated, and there is the additional runtime cost of modifying the pointers. **Pointer modification may be handled eagerly or lazily; in general, eager modification of pointers is called relocation and lazy modification is called pointer swizzling [12, 25].** Objectstore [21] is a commercial product that uses pointer relocation, Paul Wilson has developed related 266 pointer swizzling schemes [38] and other pointer relocation schemes are appearing, such as QuickStore [37] However, we argue that a significant performance advantage of a single level ....

Moss, J. *Working with Persistent Objects: To Swizzle or Not to Swizzle.* Technical Report CS 90-38, CS Department, University of Massachusetts, May 1990.

Towards a Persistence Framework for High Performance Computing.. - Nolte (1995) (Correct)

....be completely reorganized. **On** the other hand, this flexibility has its costs due to lookups in swizzle tables during the swizzling process. **A SwizzleHeap maintains these swizzling tables and may be managed according to several strategies based e.g. on eager, lazy or wavefront swizzling (see [24], 35] and [34] for exhaustive discussion of swizzling) Eager swizzling converts all addresses at once, when heap objects are relocated.** Lazy swizzling is performed upon object access, whereas wavefront swizzling is a mixture of both, swizzling all addresses in an object ahead, but moves the ....

J. E. B. Moss. *Working with Persistent Objects: To Swizzle or Not to Swizzle.* IEEE Software, 18(8), Aug. 1992.

Lumberjack: A Log-Structured Persistent Object Store - David Hulse Alan (Correct)

....into the local heap to ensure referential integrity. **At** commit time, the LONs in each object are replaced by the original PIDs before the objects are copied back to the external heap where they are made stable. **This technique of temporarily replacing PIDs with LONs is known as pointer swizzling [10].** 2.1 External Heap Architecture Within the external heap, objects are grouped within larger objects called partitions, each of which has a unique number. **An** object s PID reflects this grouping by way of its internal structure which comprises a partition number and an object number field. **The** ....

J.E.B. Moss, "*Working With Persistent Objects: To Swizzle or Not to Swizzle*", Technical Report 9038, University of Massachusetts, Massachusetts, 1991.

RT1R1/2: Report on the efficacy of Persistent Operating.. - Hulse, Dearle (Correct)

....address translation. **However**, the impact of the cost of address translation in persistent systems is not clear due to the lack of sufficient measurement. **Indeed, Moss asserts that software address translation is more efficient than utilising hardware mechanisms under conventional operating systems [34].** The utilisation of paging mechanisms suffers from two problems. **Firstly**, no operating system constructed to date provides sufficiently flexible mechanisms to exploit the hardware facilities to their full potential. **Mach** [3] and Chorus [43] for example, do provide considerable flexibility in ....

J.E.B. Moss, "*Working with Persistent Objects: To Swizzle or Not to Swizzle*", IEEE Transactions on Computers, , pp. 1-39, 1991.

Operating System Support for Persistent Systems: Past, Present .. - Dearle, Hulse (2000) (2 citations) (Correct)

....address spaces. **The** impact of the cost of address translation in persistent systems is not clear due to the lack of sufficient measurement. **For example, Moss asserts that software address translation is more efficient than utilising hardware mechanisms as exposed by conventional operating systems [45].** This may be a condemnation of the time modern operating systems take to service a trap rather than praise for software addressing techniques. **However**, since the ratio of processor speed over time required to service a trap is increasing, this is likely to become increasingly true in the ....

J.E.B. Moss, "*Working with Persistent Objects : To Swizzle or Not to Swizzle*", IEEE Transactions on Computers, pp. 1-39, 1991.

A Dynamic, Portable and Safe Approach to Byte-Code.. - Marquez, Zigman, Blackburn (Correct)

....a more lazy approach, and requires barriers over both primitive and reference fields, while the post barrier need only be applied to reference fields and is more eager. **Swizzling Strategy The choice of swizzling 6 strategy is a significant design decision for implementers of persistent systems [Moss 1992; Kemper and Kossmann 1995] and has had a major impact on the way we have approached our OPJ implementations.** For the purposes of this discussion, four broad options exist: eager swizzling, where all references are swizzled as soon as an object is faulted into memory; lazy swizzling, where ....

MOSS, J. E. B. 1992. *Working with persistent objects: To swizzle or not to swizzle*. IEEE Transactions on Software Engineering SE-18, 8 (August), 657–673.

Experience with the PerDiS large-scale data-sharing.. - Shapiro, Ferreira, Richer (2000) (Correct)

....memory by a factor of 15 (for 10K objects) The drawback is that currently the application must specify object groupings manually. **The** current implementation has known limitations. **Currently the system is very homogeneous, lacking a data translation layer, object relocation (pointer swizzling [10]) and support for the evolution of shared classes.** In order to correct these shortcomings, a system for extracting type information from binary les was developed [11] but was never integrated in the platform for lack of time. 6.2 Quantitative assessment We have done quantitative assessments of ....

J. Eliot B. Moss. *Working with persistent objects: To swizzle or not to swizzle*. IEEE Transactions on Software Engineering, 18(8):657-673, August 1992.

PerDiS: design, implementation, and use of a.. - Ferreira.. (1998) (1 citation) (Correct)

....memory. **An** object may point to any other object. **The** system ensures that pointers retain the same meaning for all application processes. **To combine persistence with real pointers while retaining exibility, many systems do swizzling, i.e. automatically translate global addresses into pointers [27,33].** PerDiS is designed to provide swizzling but the current implementation simply relies on xed address allocation [11] Starting from some persistent root pointer identi ed by a string name, a process navigates from object to object by following pointers. **For** instance, an application might ....

J. Eliot B. Moss. *Working with persistent objects: To swizzle or not to swizzle.* IEEE Transactions on Software Engineering, 18(8):657-673, August 1992.

Mostly-copying reachability-based orthogonal persistence - Hosking, Chen (1999) (1 citation) (Correct)

....barriers can represent significant overhead to the execution of any persistent program, especially if written in an orthogonally persistent language where every access might be to a persistent object. **There** are several approaches to mitigating these performance problems. **Pointer swizzling [63] is a technique that allows accesses to resident persistent objects to proceed at volatile memory hardware speeds by arranging for references to resident persistent objects to be represented as direct virtual memory addresses, as opposed to the persistent identifier format used in stable storage.** ....

MOSS, J. E. B. *Working with persistent objects: To swizzle or not to swizzle.* IEEE Trans. Softw. Eng. 18, 8 (Aug. 1992), 657--673.

Run-time support for parallel discrete event simulation languages - Wonnacott (1996) (Correct)

....doing an integer comparison) whereas in Sim locality is expressed statically at compile time. **The integral comparison could be eliminated if the APOSTLE compiler could show that the destination object is always guaranteed to be mapped to the same LP, or by using pointer swizzling techniques [Moss 92] where an OID for a local object is replaced by a pointer dynamically at run time.** The latter technique would also be useful if APOSTLE supported an object migration capability. 7.4 Performance In this section we present performance data illustrating the effect of granularity control in ....

....other using their C pointers, as in Sim (Baezner et al. 90) so that the dynamic test could be eliminated. **Conceptually** this should present no problems for the APOSTLE run time system, although such a facility has not been implemented yet. **It may also be beneficial to change or swizzle (Moss 92) the GlobalMachineIDs passed to an object (as the result of a method invocation) to just their C pointers if the objects are mapped to the same LP.** In this paper we have shown that granularity control can give large run time reductions for APOSTLE, and provided some analysis as to where ....

J. E. B. Moss, *"Working with Persistent Objects: To Swizzle or Not to Swizzle"*, IEEE Trans. on Software Engineering, August 1992, 18(2):657678.

Database Architectures - Delis, Kanitkar, Kollios (1998) (Correct)

....of an interprocess communication mechanism. **Thus**, there is a non negligible penalty involved in carrying out the above conversion in address space every time there is a reference to an object. **Instead of performing the above steps, what modern systems tend to do is to swizzle database objects [61].** In swizzling, disk based object layouts, such as tuples of certain constant length and representation, are transformed into strings of variable length. **User** applications are provided with access to these variable length strings through direct pointers. **The** key performance question in swizzling ....

J.E.B. Moss, *Working With Persistent Objects: To Swizzle or Not to Swizzle.* IEEE Transactions on Software Engineering, 18(3):103--139, 1992.

Software---Practice And Experience, Vol. 23(12).. - Making Objects.. (Correct)

No context found.

J. E. B. Moss, `*Working with persistent objects: to swizzle or not to swizzle*', COINS Technical Report, May 1990.

Object Storage Management Architectures - Biliris, Orenstein (1994) (3 citations) (Correct)

No context found.

J.E.B. Moss. *Working with persistent objects: To swizzle or not to swizzle*. IEEE Transactions on Software Engineering, 18#8#:657#673, August 1992.

Collecting Garbage in Multilevel Secure Object Stores - Bertino, Mancini, Jajodia (1994) (4 citations) (Correct)

No context found.

J.E. Moss, *Working with Persistent Objects: to Swizzle or Not to Swizzle*", IEEE Trans. on Software Engineering, Vol.18, No.8, 1992.

Implementing Crash Recovery in QuickStore: A Performance Study - White, DeWitt (1995) (12 citations) (Correct)

No context found.

J. Eliot B. Moss, "*Working with Persistent Objects: To Swizzle or Not to Swizzle*", IEEE Trans. on Software Eng., 18(8):657-673, August 1992.

First 50 documents Next 50

SWIZZLE

Pronunciation: 'swizul

Matching Terms: swizzle stick

Computing Dictionary

Definition: To convert external names, array indices, or references within a data structure into address pointers when the data str into main memory from external storage (also called "pointer swizzling"); this may be done for speed in chasing refere code (e.g. by turning lots of name lookups into pointer dereferences). The converse operation is sometimes termed "un

See also snap.

[Jargon File]

